Analisis Mitigasi Keamanan Aplikasi *Mobile* menggunakan Metode Statis dan Dinamis pada *MobSF*

Security Mitigation Analysis of Mobile Application Using Static and Dynamic Methods with MobSF

[1]Fazri Nugraha*, [2]Mansur
[1,2]Program Studi Keamanan Sistem Informasi, Jurusan Teknik Informatika, Politeknik Negeri Bengkalis
[1,2]Jl. Bathin Alam, Sungai Alam, Bengkalis, Indonesia
*e-mail: fazrinugraha795@gmail.com

(***received*:** 8 December 2025, ***revised*:** 19 December 2025, ***accepted*:** 23 December 2025)

Abstrak

Penelitian ini mengevaluasi keamanan Aplikasi *Mobile* Sistem Informasi Panen Sawit menggunakan analisis statis dan dinamis pada *Mobile Security Framework (MobSF)*. Penelitian dilatarbelakangi oleh tingginya risiko eksploitasi aplikasi berbasis *APK* dan belum adanya penilaian keamanan mendalam terhadap aplikasi yang mengelola data operasional petani. Analisis statis dilakukan untuk mendeteksi kelemahan struktural, seperti penggunaan sertifikat *debug*, *mode debugging aktif*, *minSdkVersion* rendah, serta komponen yang diekspor tanpa proteksi. Hasil awal menunjukkan *App Security Score* sebesar 43/100 (*Medium Risk*), dan setelah perbaikan konfigurasi meningkat menjadi 67/100 (*Low Risk*). Analisis dinamis kemudian digunakan untuk menilai keamanan aplikasi saat dijalankan. Pengujian menunjukkan bahwa sisi klien telah aman, dengan komunikasi terenkripsi *HTTPS* dan tanpa pencatatan data sensitif. Namun, analisis dinamis mengidentifikasi kelemahan pada sisi server, yaitu sejumlah *endpoint backend* dapat diakses tanpa autentikasi dan tanpa validasi parameter, sehingga memicu risiko *Broken Access Control* dan *Insecure Direct Object Reference (IDOR)*. Hasil penelitian menegaskan bahwa perbaikan statis efektif meningkatkan keamanan struktural aplikasi, tetapi penguatan autentikasi, otorisasi, dan validasi permintaan pada *API backend* tetap diperlukan agar keamanan aplikasi terpenuhi secara menyeluruh sebelum diterapkan pada lingkungan operasional. Berbeda dari penelitian sebelumnya yang umumnya berfokus pada pemetaan kerentanan, penelitian ini mengevaluasi efektivitas mitigasi keamanan secara bertahap melalui peningkatan skor analisis statis dan validasi ulang menggunakan analisis dinamis, sehingga memberikan gambaran keamanan aplikasi *mobile* yang mencakup sisi klien dan backend secara komprehensif.

Kata kunci: *API backend*, analisis dinamis, keamanan aplikasi *mobile*, *mobile security framework (MobSF)*, analisis statis

Abstract

This study evaluates the security of the Mobile Application for the Palm Oil Harvest Information System using static and dynamic analysis through the Mobile Security Framework (MobSF). The research is motivated by the high risk of exploitation in APK-based applications and the lack of in-depth security assessments for applications that manage farmers' operational data. Static analysis was conducted to identify structural weaknesses, including the use of debug certificates, enabled debugging mode, a low minimum SDK version (minSdkVersion), and exported components without proper protection. The initial results showed an App Security Score of 43/100 (Medium Risk), which increased to 67/100 (Low Risk) after configuration improvements were applied. Dynamic analysis was then performed to assess application security during runtime. The results indicated that the client side was relatively secure, with HTTPS-encrypted communication and no logging of sensitive data. However, dynamic analysis revealed vulnerabilities on the server side, where several backend endpoints could be accessed without authentication and without parameter validation, leading to potential risks of Broken Access Control and Insecure Direct Object Reference (IDOR). The findings confirm that static improvements are effective in strengthening the structural security of the

 ISSN:2302-8149
e-ISSN:2540-9719

application. Nevertheless, reinforcing authentication, authorization, and request validation mechanisms on the backend API remains essential to ensure comprehensive security before deployment in an operational environment. Unlike previous studies that generally focus only on vulnerability mapping, this study evaluates the effectiveness of security mitigation in a step-by-step manner by demonstrating improvements in static analysis scores and re-validating the results through dynamic analysis. Therefore, this research provides a more comprehensive security assessment of mobile applications by covering both client-side and backend aspects.

Keywords: *API backend, dynamic analysis, mobile application security, mobile security framework (MobSF), static analysis*

1 Pendahuluan

Pemanfaatan perangkat *mobile* sebagai media utama akses digital meningkat secara signifikan dalam beberapa tahun terakhir. Di Indonesia, 98,7% masyarakat berusia di atas 16 tahun mengakses internet melalui ponsel [1]. Data terbaru dari Survey Internet Indonesia 2024 yang dilakukan oleh APJII menunjukkan bahwa sebanyak 99,51% pengguna internet mengakses layanan digital melalui ponsel, menjadikan *Android* sebagai platform dominan dalam penyediaan aplikasi operasional di berbagai sektor [2].

Penerapan aplikasi mobile dengan integrasi geolokasi dan data *real-time* terbukti meningkatkan efisiensi operasional, mempercepat pencarian informasi hingga 40–60%, serta mencapai tingkat kepuasan pengguna hingga 94% [3]. Perkembangan tersebut turut mendorong adopsi teknologi *mobile* pada industri kelapa sawit sebagai salah satu komoditas strategis nasional [4]. Digitalisasi proses operasional, termasuk pencatatan hasil panen dan pendataan aktivitas lapangan, menjadi kebutuhan penting untuk meningkatkan akurasi dan efisiensi kerja [5]. Sejalan dengan itu, dikembangkan aplikasi mobile Sistem Informasi Panen Sawit (sawitgodigi.apk versi 1.0.0) untuk mendukung pencatatan hasil panen, pemesanan penjemputan, serta pemantauan aktivitas lapangan melalui perangkat *Android*. Aplikasi ini digunakan oleh petani dan supir untuk mempercepat proses pencatatan dan penjemputan hasil panen sawit. Namun, karena aplikasi ini menangani data strategis, aspek keamanan perlu menjadi perhatian utama untuk menjamin kerahasiaan, keutuhan, dan ketersediaan informasi dari akses maupun modifikasi oleh pihak yang tidak berwenang [6].

Di sisi lain, ancaman terhadap aplikasi *mobile* terus meningkat. Laporan *Zimperium Global Mobile Threat Report* 2025 menemukan bahwa lebih dari 25% perangkat *mobile* masih menggunakan sistem operasi tanpa pembaruan keamanan, sementara distribusi aplikasi melalui mekanisme *sideloading* berpotensi membuka peluang penyisipan *malware*, eksploitasi izin berlebih, dan manipulasi kode aplikasi [7]. Selain itu, sekitar 34% aplikasi *Android* tidak memiliki perlindungan dasar seperti *obfuscation*, enkripsi lokal, dan proteksi *runtime*, sehingga rentan terhadap *reverse engineering* dan modifikasi tidak sah [7]. Kondisi ini menunjukkan bahwa aplikasi yang didistribusikan di luar *Google Play Store*, termasuk file *APK*, memerlukan pengujian keamanan yang menyeluruh.

Mobile Security Framework (MobSF) menjadi salah satu alat yang banyak digunakan untuk menguji keamanan aplikasi *Android* karena mampu mengintegrasikan dua jenis pengujian utama, yaitu analisis statis dan analisis dinamis [8]. Analisis statis berfokus pada pemeriksaan struktur internal aplikasi tanpa menjalankannya, termasuk penilaian izin, konfigurasi *manifest*, dan pustaka pihak ketiga [9][10]. Sementara itu, analisis dinamis dilakukan dengan menjalankan aplikasi pada lingkungan *emulator* yang telah di-root untuk mengamati perilaku *runtime*, pemanggilan *API*, serta keamanan komunikasi jaringan [8][11]. Sejumlah penelitian terdahulu telah mengimplementasikan *MobSF* untuk menganalisis aplikasi publik seperti *Mobile* JKN, aplikasi *e-commerce*, dan aplikasi transportasi online, dan berhasil mengidentifikasi kelemahan seperti *weak cryptography*, *hardcoded secrets*, dan *SQL injection* [12][13], [14]. Namun, sebagian besar penelitian tersebut hanya menekankan pemetaan kerentanan tanpa mengevaluasi efektivitas perbaikan yang diterapkan.

Kesenjangan tersebut menjadi dasar pentingnya penelitian ini, terlebih karena aplikasi Sistem Informasi Panen Sawit yang dikembangkan masih berada pada tahap pra-produksi dan didistribusikan secara informal dalam bentuk *APK*, sehingga rentan terhadap modifikasi. Oleh karena itu, diperlukan sebuah pendekatan yang tidak hanya mengidentifikasi kerentanan, tetapi juga melakukan perbaikan berdasarkan hasil analisis statis serta mengevaluasi dampaknya melalui pengujian dinamis.

 ISSN:2302-8149
e-ISSN:2540-9719

Berdasarkan permasalahan tersebut, penelitian ini berupaya menjawab dua fokus utama, yaitu bagaimana pendekatan analisis statis dan analisis dinamis pada *MobSF* diterapkan dalam menguji keamanan aplikasi *mobile* Sistem Informasi Panen Sawit, serta bagaimana perbaikan yang dilakukan setelah analisis statis memengaruhi hasil evaluasi keamanan pada tahap analisis dinamis. Penelitian ini bertujuan untuk melakukan analisis keamanan aplikasi *mobile* Sistem Informasi Panen Sawit menggunakan pendekatan analisis statis dan dinamis pada *MobSF*, serta mengevaluasi hasil kedua analisis tersebut untuk memperoleh gambaran menyeluruh mengenai kondisi keamanan aplikasi.

Berbeda dari penelitian sebelumnya yang umumnya hanya berfokus pada pemetaan kerentanan aplikasi mobile menggunakan *MobSF*, penelitian ini menerapkan evaluasi keamanan secara bertahap yang mencakup identifikasi kerentanan, penerapan mitigasi berbasis analisis statis, serta validasi ulang melalui analisis dinamis. Pendekatan ini memungkinkan penilaian efektivitas mitigasi yang diterapkan sekaligus memberikan gambaran keamanan aplikasi *mobile* yang lebih komprehensif dengan mempertimbangkan sisi klien dan backend secara bersamaan.

2 Tinjauan Literatur

Sejumlah penelitian sebelumnya menunjukkan bahwa aplikasi *Android* masih memiliki kerentanan signifikan yang dapat dieksploitasi. Salah satu penelitian yang memanfaatkan analisis statis dan dinamis menilai keamanan aplikasi layanan administrasi desa dan menemukan berbagai kerentanan penting, seperti penggunaan algoritma kriptografi usang (*MD5* dan *SHA-1*), keberadaan *hardcoded secrets*, izin berbahaya, serta ketiadaan deteksi perangkat *root* dan *debugger*. Meskipun telah menerapkan *SSL Pinning*, pengujian dinamis memperlihatkan bahwa mekanisme tersebut masih dapat dilewati, sehingga aplikasi rentan terhadap manipulasi saat dijalankan [15]. Pendekatan serupa juga diterapkan pada aplikasi layanan kesehatan nasional yang mengelola data sensitif pengguna. Hasil analisis statis menunjukkan adanya penggunaan kriptografi lemah, izin akses yang tidak proporsional, serta temuan *hardcoded secrets*. Pengujian dinamis mengungkap bahwa fitur keamanan seperti *root detection*, pengecekan *debug*, dan *SSL Pinning* masih dapat dilewati, serta terdapat potensi serangan seperti *Janus* dan *SQL Injection*. Hal ini menunjukkan bahwa meskipun aplikasi digunakan pada skala nasional, perlindungan yang diterapkan belum memadai [11]. Penelitian selanjutnya pada aplikasi *mobile* commerce mengintegrasikan *MobSF* dengan pedoman *OWASP MASTG* untuk memperoleh hasil yang lebih komprehensif. Temuannya menunjukkan masih adanya kelemahan terkait kebijakan autentikasi yang tidak kuat, penyimpanan data sensitif tanpa perlindungan, risiko *brute force*, serta konfigurasi internal aplikasi yang memungkinkan eksploitasi. Gabungan analisis statis dan dinamis memberikan gambaran yang lebih jelas mengenai bagaimana kelemahan pada struktur aplikasi dapat memengaruhi keamanan saat dijalankan [16].

Selain itu, terdapat penelitian yang fokus pada analisis statis terhadap aplikasi berbasis *Flutter*. Studi tersebut menemukan bahwa aplikasi berada pada kategori risiko tinggi dengan skor awal 37/100 akibat pengaturan teknis yang tidak aman, seperti penggunaan *debug certificate*, aktifnya *debuggable mode*, versi *SDK* yang belum diperbarui, serta tidak digunakannya *release keystore*. Setelah dilakukan perbaikan teknis, skor keamanan meningkat menjadi 61/100 dan masuk kategori risiko rendah. Hasil ini menegaskan bahwa analisis statis *MobSF* efektif digunakan sebagai dasar peningkatan keamanan sebelum aplikasi dirilis, meskipun masih diperlukan analisis dinamis untuk memastikan kondisi *runtime* aplikasi [17]. Pada aplikasi HIMFO, analisis statis menunjukkan penggunaan algoritma kriptografi lemah, celah *SSL bypass*, izin berbahaya, serta beberapa referensi ke domain eksternal yang dinilai berisiko. Meskipun aplikasi telah menerapkan deteksi *root*, sejumlah kerentanan teknis masih ditemukan sehingga diperlukan pengujian lanjutan sebelum aplikasi dirilis kembali [18]. Sementara itu, penelitian lain pada aplikasi layanan publik mengidentifikasi kelemahan serupa termasuk algoritma kriptografi usang, kebocoran informasi internal, dan ketiadaan mekanisme pencegahan manipulasi yang menyebabkan aplikasi berada pada kategori skor keamanan rendah dan membutuhkan perbaikan menyeluruh [19]. Studi komparatif terhadap aplikasi transportasi online memperlihatkan pola kerentanan yang serupa, di mana seluruh aplikasi yang diuji memiliki izin akses berlebih, kriptografi lemah, serta potensi *bypass SSL*. Tidak adanya penerapan deteksi *root* pada aplikasi-aplikasi tersebut semakin meningkatkan risiko eksploitasi, terutama karena aplikasi tersebut menangani data lokasi secara *real-time* dan digunakan oleh pengguna dalam jumlah besar [13].

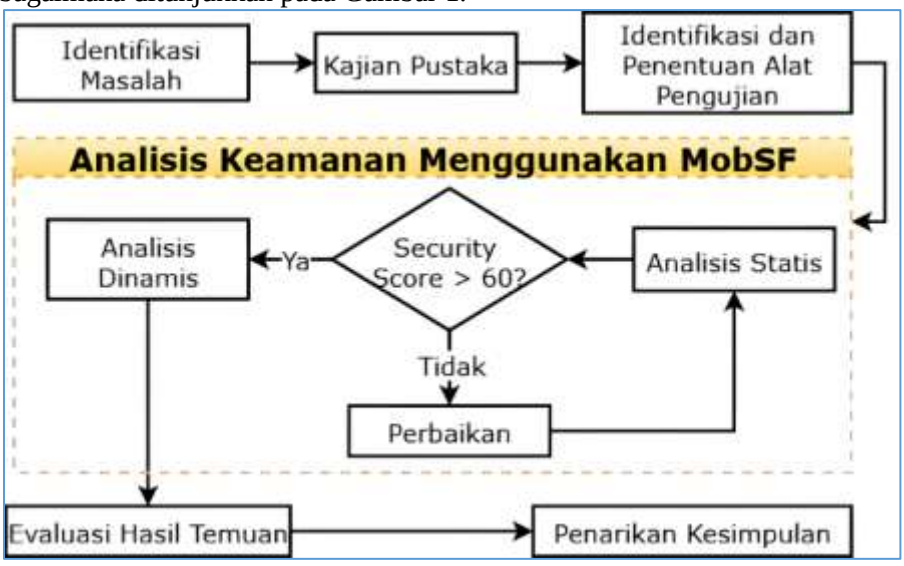
 ISSN:2302-8149
e-ISSN:2540-9719

Selain analisis statis, penelitian lain menekankan pentingnya pengujian dinamis untuk mendeteksi kerentanan yang hanya muncul ketika aplikasi dijalankan pada perangkat *rooted* atau *emulator*. Pengujian ini memungkinkan pengamatan langsung terhadap manipulasi lalu lintas jaringan, injeksi input *runtime*, serta *bypass SSL* yang tidak dapat terdeteksi melalui analisis statis saja. Hal ini menunjukkan bahwa kedua metode pengujian diperlukan untuk memahami kondisi keamanan aplikasi secara menyeluruh [14]. Dari keseluruhan penelitian terdahulu, tampak bahwa kerentanan aplikasi *Android* cenderung berulang pada berbagai kategori aplikasi. Namun, fokus penelitian umumnya masih terbatas pada sisi klien aplikasi, sementara aspek keamanan *backend* belum banyak dievaluasi secara mendalam. Selain kelemahan pada sisi klien, penelitian terkini menunjukkan bahwa kerentanan keamanan aplikasi *mobile* banyak muncul pada lapisan *backend*, khususnya pada layanan *Application Programming Interface (API)*. Meskipun komunikasi telah menggunakan protokol *HTTPS*, kelemahan kontrol akses pada *backend* masih sering ditemukan dan memicu terjadinya *Broken Access Control* serta *Insecure Direct Object Reference (IDOR)* [20]. Kerentanan tersebut umumnya disebabkan oleh ketiadaan verifikasi kepemilikan objek pada parameter permintaan *API*, sehingga memungkinkan pengguna mengakses data milik entitas lain melalui manipulasi nilai identitas pada *URL* atau *payload* [21].

Meskipun sejumlah penelitian telah menggunakan *MobSF* dalam analisis statis maupun dinamis, sebagian besar hanya berfokus pada pemetaan kerentanan tanpa mengevaluasi efektivitas perbaikan yang dilakukan. Dengan demikian, penelitian ini mengisi kekosongan tersebut melalui penerapan analisis keamanan dua tahap, yaitu identifikasi kerentanan melalui analisis statis, diikuti perbaikan dan evaluasi ulang menggunakan analisis dinamis. Prosedur ini memungkinkan verifikasi langsung terhadap efektivitas mitigasi dan memberikan gambaran keamanan aplikasi yang lebih komprehensif dibandingkan penelitian sebelumnya.

3 Metode Penelitian

Penelitian ini menggunakan *Mobile Security Framework (MobSF)* untuk melakukan menganalisis keamanan terhadap Aplikasi *Mobile* Sistem Informasi Panen Sawit melalui dua pendekatan utama, yaitu analisis statis dan analisis dinamis. Tahapan penelitian disusun secara sistematis sebagaimana ditunjukkan pada Gambar 1.



Gambar 1 Tahapan penelitian

Penelitian ini mengikuti tahapan terstruktur sebagaimana pada gambar 1. Proses dimulai dengan identifikasi masalah dan kajian pustaka terkait analisis statis dan dinamis menggunakan *MobSF*. Setelah menetapkan perangkat yang digunakan, dilakukan analisis statis untuk memperoleh *security score* awal. Jika skor belum mencapai batas 60%, aplikasi diperbaiki dan diuji ulang. Setelah memenuhi kriteria, analisis dinamis dilakukan untuk mengamati perilaku aplikasi. Hasil kedua analisis kemudian dievaluasi untuk menilai tingkat keamanan aplikasi, dan penelitian ditutup dengan penyusunan kesimpulan serta rekomendasi.

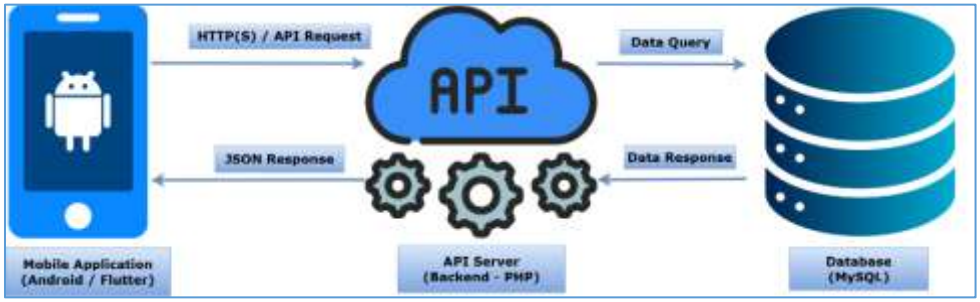
 ISSN:2302-8149
e-ISSN:2540-9719

3.1 Konfigurasi Lingkungan Analisis

Dalam pengujian aplikasi Sistem Informasi Panen Sawit, lingkungan analisis disiapkan dengan memadukan perangkat keras dan perangkat lunak pendukung pengujian keamanan. Analisis statis dan dinamis dilakukan menggunakan *MobSF* versi 4.4.3 yang dijalankan melalui *Docker Desktop* versi 4.41.2. Pengujian dinamis dilakukan pada *emulator Genymotion* versi 3.9.0 dengan perangkat virtual *Google Pixel* 3 berbasis *Android* 11 (*API* Level 30) yang dijalankan di atas *VirtualBox* versi 7.0. Proses instrumentasi *runtime* menggunakan *Frida* yang terintegrasi secara bawaan (*embedded*) pada modul analisis dinamis *MobSF*. Proses peninjauan struktur internal aplikasi dilakukan menggunakan *Visual Studio Code* versi 1.107.0, sedangkan aplikasi *mobile* Sistem Informasi Panen Sawit (sawitgodigi.apk versi 1.0.0) digunakan sebagai objek pengujian. Seluruh pengujian dilakukan pada sistem operasi *Windows* 11 Pro 64-bit.

3.2 Arsitektur Sistem dan Ruang Lingkup Pengujian

Aplikasi Sistem Informasi Panen Sawit merupakan aplikasi *mobile* berbasis *Android* yang dikembangkan menggunakan *framework Flutter* dan berperan sebagai klien. Aplikasi *mobile* berkomunikasi dengan sistem *backend* melalui layanan *Application Programming Interface (API)* menggunakan protokol *HTTP(S)*. *Backend* aplikasi diimplementasikan pada *web server hosting* dalam bentuk layanan *API* berbasis *PHP* dan terhubung dengan basis data *MySQL* sebagai media penyimpanan data.



Gambar 2 Arsitektur sistem aplikasi *mobile* sistem informasi panen sawit

Gambar 2 menunjukkan arsitektur sistem aplikasi yang menerapkan pola *client–server*, di mana aplikasi *mobile Android* berperan sebagai klien yang berkomunikasi dengan *API* server sebagai lapisan logika aplikasi melalui protokol *HTTP(S)*, serta terhubung dengan basis data *MySQL* sebagai media penyimpanan data. Pengujian keamanan pada sisi server dalam penelitian ini dilakukan dari perspektif aplikasi *mobile*. Analisis *backend* tidak mencakup pemeriksaan kode sumber maupun konfigurasi internal *server* secara langsung, melainkan dilakukan dengan mengamati interaksi antara aplikasi *mobile* dan *endpoint API backend* selama proses *runtime* menggunakan analisis dinamis pada *MobSF*.

3.3 Tahapan Analisis Statis *MobSF*

Analisis statis dilakukan menggunakan *MobSF* untuk mengevaluasi struktur internal aplikasi tanpa melakukan eksekusi program. Proses dimulai dengan menyiapkan file *APK* dan mengunggahnya ke *MobSF* setelah seluruh lingkungan pengujian terkonfigurasi. Pada tahap pemindaian, *MobSF* melakukan ekstraksi terhadap komponen aplikasi, termasuk *AndroidManifest.xml*, konfigurasi izin, pustaka pihak ketiga, serta komponen yang diekspor. Hasil analisis kemudian disajikan dalam bentuk laporan keamanan yang memuat daftar kerentanan, tingkat keparahan, serta rekomendasi mitigasi. Selain itu, *MobSF* menghasilkan *security score* sebagai indikator tingkat keamanan aplikasi, dengan klasifikasi sebagaimana ditampilkan pada Tabel 1 [22].

Tabel 1 Klasifikasi *security score* analisis statis *MobSF*

Security Score	Kategori Risiko	Grade
≥ 60%	Low Risk	A
40%-59%	Medium Risk	B
30%-39%	High Risk	C
< 30%	Critical	F

 ISSN:2302-8149
e-ISSN:2540-9719

Pada penelitian ini apabila nilai *security score* berada di bawah ambang batas 60%, aplikasi dinyatakan belum memenuhi standar keamanan struktural dan harus diperbaiki sesuai rekomendasi yang dihasilkan *MobSF*. Setelah perbaikan diterapkan, file *APK* dipindai kembali hingga memperoleh nilai yang memenuhi kriteria keamanan. Tahap analisis statis dinyatakan selesai ketika aplikasi mencapai kategori risiko rendah dan dapat dilanjutkan ke proses analisis dinamis.

3.4 Tahapan Analisis Dinamis *MobSF*

Analisis dinamis dilakukan setelah aplikasi dinyatakan aman pada tahap statis. Pada tahap ini, file *APK* dijalankan di dalam *emulator Android* yang telah di *root* menggunakan modul analisis dinamis pada *MobSF*. Pengujian dilakukan secara *runtime* untuk mengamati perilaku aplikasi ketika digunakan secara langsung. Selama pengujian, penguji melakukan interaksi manual pada aplikasi, seperti membuka menu, mengisi formulir, dan mengakses fitur utama. *MobSF* kemudian memantau aktivitas *runtime*, meliputi lalu lintas jaringan, penggunaan izin, pemanggilan *API* sistem, serta potensi akses data sensitif atau indikasi manipulasi seperti *debugging* maupun deteksi *emulator*. Setelah proses interaksi selesai, *MobSF* menghasilkan laporan *runtime security assessment* yang berisi log aktivitas, pengujian koneksi jaringan, dan potensi ancaman.

3.5 Evaluasi Hasil Temuan

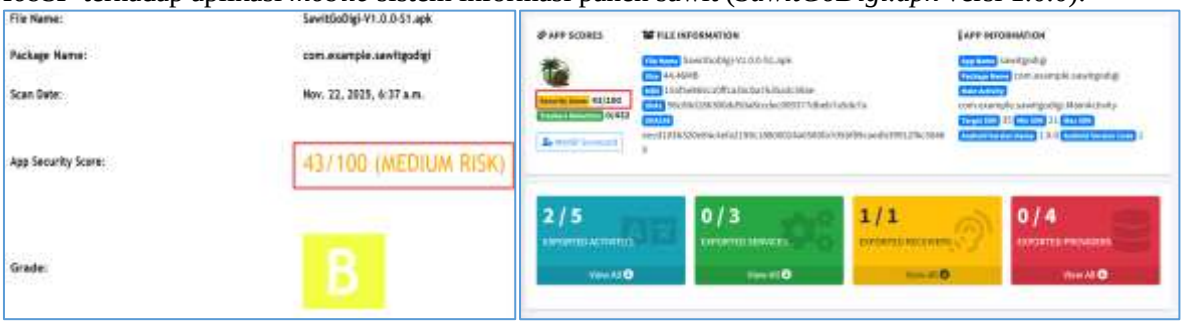
Evaluasi hasil temuan dilakukan dengan mengompilasi dan membandingkan keluaran analisis statis dan analisis dinamis untuk memperoleh gambaran komprehensif mengenai kondisi keamanan aplikasi.

4 Hasil dan Pembahasan

Bagian ini menyajikan hasil analisis keamanan aplikasi *mobile* Sistem Informasi Panen Sawit menggunakan *MobSF*. Pembahasan difokuskan pada hasil pemindaian, perbaikan kerentanan, serta evaluasi keamanan aplikasi pada sisi klien dan *backend.*

4.1 Hasil Analisis Statis *MobSF*

Gambar 3 menampilkan hasil pemindaian awal analisis statis yang dilakukan *menggunakan MobSF* terhadap aplikasi *mobile* sistem informasi panen sawit (*SawitGoDigi.apk* versi 1.0.0).



Gambar 3 Hasil analisis statis *MobSF*

Berdasarkan hasil yang ditunjukkan pada gambar 3, aplikasi memperoleh *App Security Score* sebesar 43/100, yang dikategorikan sebagai *Medium Risk*. Sejumlah temuan berisiko *High* juga teridentifikasi, seperti penggunaan *debug certificate*, mode *debug* aktif, dukungan instalasi pada perangkat *Android* versi lama, serta komponen aplikasi yang diekspor tanpa proteksi. Rincian lengkap hasil temuan kerentanan dari analisis statis *MobSF* dapat dilihat pada Tabel 2.

Tabel 2 Temuan kerentanan berdasarkan analisis statis *MobSF*

No	Jenis Temuan	Tingkat Risiko	Uraian Temuan
1	*Application signed with debug certificate*	*High*	Aplikasi masih menggunakan sertifikat *debug* yang hanya ditujukan untuk proses pengembangan sehingga tidak memenuhi standar keamanan untuk rilis produksi.
2	*App can be installed on a vulnerable unpatched Android version (minSdk=21)*	*High*	Aplikasi dapat dipasang pada perangkat *Android* versi lama yang memiliki banyak celah keamanan dan tidak lagi menerima pembaruan sistem.

 ISSN:2302-8149
e-ISSN:2540-9719

No	Jenis Temuan	Tingkat Risiko	Uraian Temuan
3	*Debug enabled for app [android:debuggable=true]*	*High*	Mode *debug* masih aktif sehingga memungkinkan pihak lain melakukan analisis mendalam atau memodifikasi jalannya aplikasi.
4	*Application vulnerable to Janus vulnerability*	*Warning*	Skema penandatanganan v1 yang digunakan masih rentan terhadap serangan Janus pada beberapa versi *Android*.
5	*Application data can be backed up [android:allowBackup] flag is missing*	*Warning*	Pengaturan pencadangan data belum dibatasi sehingga data aplikasi berpotensi diekstraksi melalui mekanisme *backup*.
6	*Activity(com.google.firebase.auth. internal.GenericIdpActivity) is not Protected. [android:exported=true]*	*Warning*	Aktivitas autentikasi *Firebase* ini diekspos dan dapat diakses aplikasi lain karena status *exported=true*.
7	*Activity(com.google.firebase.auth. internal.RecaptchaActivity) is not Protected. [android:exported=true]*	*Warning*	Aktivitas *reCAPTCHA* juga terbuka untuk aplikasi lain sehingga memerlukan kontrol akses tambahan.
8	*Broadcast Receiver (androidx. profileinstaller.ProfileInstallReceive r) is Protected by a permission, but the protection level of the permission should be checked.*	*Warning*	*BroadcastReceiver* ini menerima intent dari aplikasi lain dan membutuhkan verifikasi izin agar tidak disalahgunakan.
9	*App creates temp file. Sensitive information should never be written into a temp file.*	*Warning*	Aplikasi membuat file sementara yang dapat menyimpan data tertentu sehingga perlu dikelola agar tidak diekspos.
10	*App can read/write to External Storage. Any App can read data written to External Storage.*	*Warning*	Aplikasi dapat membaca dan menulis ke penyimpanan eksternal yang dapat diakses oleh aplikasi lain sehingga data rentan dimodifikasi.
11	*Files may contain hardcoded sensitive information like usernames, passwords, keys etc.*	*Warning*	Ditemukan potensi informasi sensitif seperti kredensial atau kunci yang tertanam langsung di dalam kode aplikasi.
12	*This app may contain hardcoded secrets*	*Warning*	Aplikasi berpotensi menyimpan secret seperti token atau *API key* secara langsung pada kode sehingga mudah diekstraksi.
13	*The App logs information. Sensitive information should never be logged.*	Info	Aplikasi masih mencatat informasi ke dalam log, yang dapat memuat data sensitif bila perangkat sedang dalam mode *debugging*.
14	*App can write to App Directory. Sensitive Information should be encrypted.*	Info	Penyimpanan internal aplikasi belum didukung enkripsi tambahan sehingga data di dalamnya lebih rentan jika perangkat dikompromikan.
15	*This App copies data to clipboard. Sensitive data should not be copied to clipboard as other applications can access it.*	Info	Data yang disalin ke *clipboard* dapat diakses aplikasi lain karena *clipboard* bersifat global pada sistem *Android*.

Temuan pada Tabel 2 menunjukkan bahwa aplikasi masih memiliki kelemahan keamanan yang cukup signifikan. Sebagian besar kerentanan berada pada kategori *High* dan *Warning*, terutama terkait konfigurasi penandatanganan aplikasi, mode *debugging*, kompatibilitas sistem operasi, serta komponen aplikasi yang diekspor tanpa proteksi.

4.2 Perbaikan Kerentanan Hasil Analisis Statis

Berdasarkan hasil pemindaian awal *MobSF*, ditemukan sejumlah kerentanan dengan tingkat risiko *High* dan *Warning* sehingga diperlukan serangkaian perbaikan pada berkas *build.gradle*, *AndroidManifest.xml*, serta konfigurasi *keystore*. Seluruh perbaikan kemudian diverifikasi melalui pemindaian ulang hingga aplikasi mencapai skor keamanan yang memenuhi standar minimal.

1) Menonaktifkan Mode *Debug* (temuan: *Debug enabled for app [android:debuggable=true]*)

Sebelum perbaikan :

```
    buildTypes {
        release {
```

 ISSN:2302-8149
e-ISSN:2540-9719

```
            signingConfig = signingConfigs.getByName("debug")
            isDebuggable = true
        }
    }
```

Sesudah perbaikan :

```
    buildTypes {
        release {
            signingConfig = signingConfigs.getByName("debug")
        }
    }
```

Penghapusan atribut *isDebuggable* pada *release build* memastikan aplikasi tidak berjalan dalam mode *debug* saat rilis, sehingga mengurangi risiko eksploitasi melalui debugging maupun *hooking* pada saat *runtime*.

2) Mengganti *Debug Certificate dengan Release Keystore* (temuan: *Application signed with debug certificate*)

Sebelum perbaikan : aplikasi menggunakan sertifikat debug bawaan.

Sesudah perbaikan :

```
    // Mengimpor konfigurasi dari key.properties
    def keystoreProperties = new Properties()
    def keystorePropertiesFile = rootProject.file("key.properties")
    if (keystorePropertiesFile.exists()) {
        keystoreProperties.load(new FileInputStream(keystorePropertiesFile))
    }
    android {
        signingConfigs {
            release {
                keyAlias = keystoreProperties['keyAlias']
                keyPassword = keystoreProperties['keyPassword']
                storeFile = file(keystoreProperties['storeFile'])
                storePassword = keystoreProperties['storePassword']
            }
        }
        buildTypes {
            release {
                signingConfig = signingConfigs.release
            }
        }
    }
```

Konfigurasi ini memindahkan proses penandatanganan aplikasi dari *debug certificate* ke *release keystore* yang didefinisikan dalam berkas *key.properties*. Dengan demikian, identitas aplikasi menjadi unik dan lebih sulit dipalsukan, sehingga kerentanan "*Application signed with debug certificate*" tidak lagi muncul pada pemindaian lanjutan.

3) Meningkatkan *minSdkVersion* (temuan: *App can be installed on a vulnerable unpatched Android version* (*minSdk*=21))

Sebelum perbaikan :

```
    minSdk = flutter.minSdkVersion
```

Sesudah perbaikan :

```
    minSdk = 29
```

Peningkatan *minSdk* ke *API* level 29 membatasi aplikasi hanya dapat dipasang pada perangkat dengan sistem operasi yang lebih baru dan relatif lebih aman, sehingga mengurangi risiko eksploitasi yang berasal dari kelemahan sistem operasi lama.

 ISSN:2302-8149
e-ISSN:2540-9719

4) Menonaktifkan Fitur *Backup* Data (temuan: *Application data can be backed up* [*android:allowBackup*] *flag is missing*)

Sebelum perbaikan :

```
<application
  ….tanpa menonaktifkan fitur backup data ….>
```

Sesudah perbaikan :

```
<application
  …….android:allowBackup="false">
```

Penambahan atribut *android:allowBackup="false"* menonaktifkan pencadangan data aplikasi melalui *backup* standar, sehingga menutup potensi kebocoran data sensitif yang tersimpan di direktori internal aplikasi.

5) Penghapusan Komponen *Firebase* dan *Receiver* yang Diekspor (temuan: aktivitas dan *receiver Firebase* berstatus *exported*)

Sebelum perbaikan :

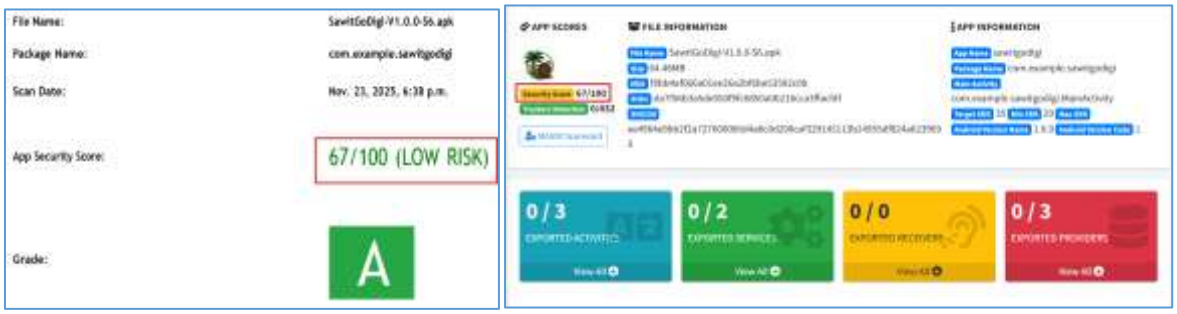
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
  ......</manifest>
dependencies {
  implementation(platform("com.google.firebase:firebase-bom:32.2.2"))
  implementation("com.google.firebase:firebase-auth-ktx")
}
```

Sesudah perbaikan :

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools">
  <receiver
    android:name="androidx.profileinstaller.ProfileInstallReceiver"
     tools:node="remove" />
</manifest>
dependencies {
}
```

Penghapusan dependensi *Firebase* dari *build.gradle* serta penambahan instruksi *tools:node="remove"* pada *ProfileInstallReceiver* memastikan bahwa komponen-komponen tersebut tidak lagi disertakan dan diekspor dalam berkas *manifest* akhir.

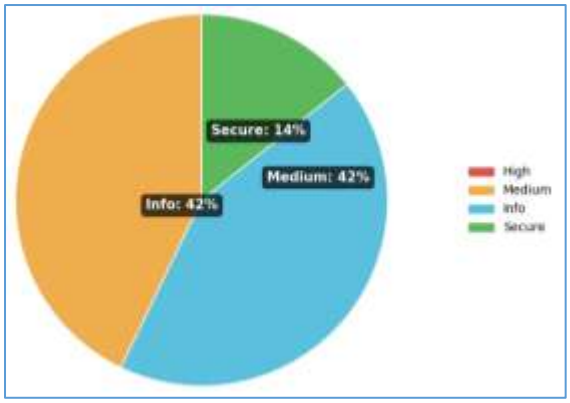
4.3 Hasil Pemindaian Akhir Analisis Statis *MobSF*



Gambar 4 Hasil akhir analisis statis *MobSF*

Gambar 4 menunjukkan hasil pemindaian akhir menggunakan *MobSF*, yang dilakukan setelah seluruh perbaikan kerentanan diterapkan pada berkas *build.gradle*, *AndroidManifest.xml*, serta *konfigurasi keystore*. Aplikasi memperoleh *App Security Score* sebesar 67/100, sehingga masuk dalam kategori *Low Risk*, yang menandakan bahwa aplikasi telah memenuhi standar keamanan struktural dan layak untuk melanjutkan ke tahap analisis dinamis. Aplikasi tidak lagi menunjukkan kerentanan dengan tingkat risiko *High*, dan seluruh temuan kritis yang ditemukan pada pemindaian awal telah berhasil diatasi.

 ISSN:2302-8149
e-ISSN:2540-9719



Gambar 5 Severity distribution

Gambar 5 menyajikan distribusi tingkat risiko pada pemindaian akhir, yaitu 42% temuan berada pada kategori Medium, 42% pada kategori Info, dan 14% pada kategori Secure, tanpa temuan berisiko tinggi. Temuan yang tersisa berada pada kategori *Warning* dan Info, yang umumnya berasal dari komponen bawaan *framework* atau *library* dan tidak berdampak langsung pada integritas keamanan inti aplikasi.

Perbaikan yang dilakukan pada setiap tahap pemindaian menunjukkan peningkatan keamanan aplikasi secara bertahap. Jumlah kerentanan berisiko tinggi menurun pada setiap siklus, dan kategori temuan bergeser menuju tingkat risiko yang lebih rendah. Ringkasan perubahan skor keamanan serta kerentanan yang berhasil diperbaiki pada setiap tahap ditampilkan pada Tabel 3.

Tabel 3 Hasil scan peningkatan *security score* aplikasi

Pemindaian	*Security Score*	*High*	*Warning*	Info	Kerentanan yang Diperbaiki	Hasil Perbaikan
Pemindaian 1	43/100	3	9	3	Belum ada perbaikan	Kerentanan kritis teridentifikasi sebagai dasar perbaikan
Pemindaian 2	47/100	2	10	3	*Debug enabled for app*	Kerentanan *debugging* hilang, risiko sedikit menurun
Pemindaian 3	51/100	1	9	3	*Application signed with debug certificate*	Identitas aplikasi aman, kerentanan sertifikat hilang
Pemindaian 4	57/100	0	8	3	*App can be installed on vulnerable unpatched Android version*	Semua kerentanan High hilang
Pemindaian 5	58/100	0	7	3	*Application data can be backed up*	Data aplikasi tidak dapat dicadangkan
Pemindaian 6	67/100	0	3	3	*Activity Firebase exported,ProfileInstallReceiver, hardcoded secrets dari library Firebase*	Aplikasi mencapai Low Risk, siap analisis dinamis

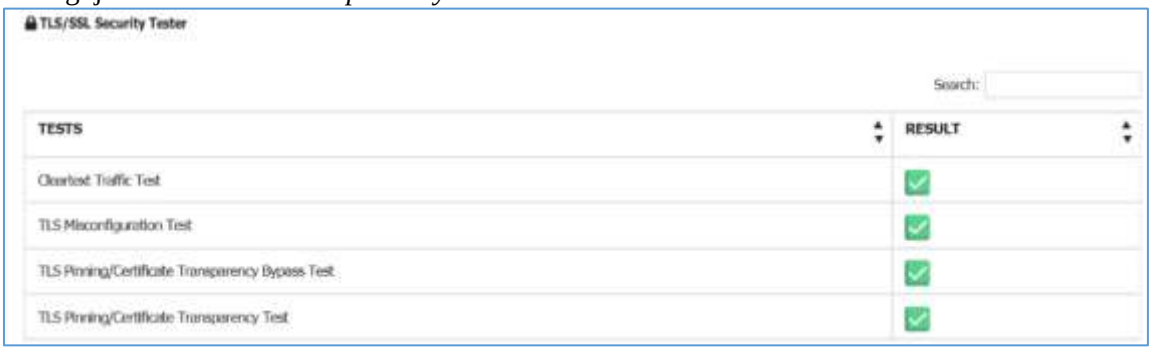
Tabel 3 menunjukkan bahwa peningkatan keamanan berlangsung melalui lima kali perbaikan dari enam pemindaian. Pada pemindaian awal, aplikasi memperoleh skor 43/100 dengan tiga temuan *High Risk* sebagai dasar identifikasi kerentanan. Perbaikan yang dilakukan pada pemindaian berikutnya secara konsisten meningkatkan skor keamanan dan mengurangi temuan kritis. Perbaikan yang diterapkan mencakup penghapusan mode *debugging*, penyelesaian penggunaan *debug certificate*, penghentian instalasi pada versi *Android* rentan, penonaktifan fitur *backup*, serta perbaikan komponen *Firebase* dan penghapusan *hardcoded secrets*. Rangkaian perbaikan ini menghilangkan seluruh temuan *High Risk* pada Pemindaian 4 dan meningkatkan skor hingga 67/100 pada pemindaian 6, menempatkan aplikasi pada kategori *Low Risk*. Sesuai alur penelitian, capaian skor di atas 60% menjadi indikator bahwa aplikasi telah memenuhi syarat untuk melanjutkan ke tahap analisis dinamis.

 ISSN:2302-8149
e-ISSN:2540-9719

4.4 Hasil Analisis Dinamis *MobSF*

Analisis dinamis dilakukan untuk mengevaluasi perilaku aplikasi saat dijalankan dalam kondisi runtime. Pengujian dilakukan pada *emulator Android* yang telah di *root*, dengan pemantauan melalui *TLS/SSL Security Tester*, *API Monitor*, *instrumentasi Frida*, serta *Logcat Runtime Logs*. Seluruh fitur utama aplikasi autentikasi, pemuatan profil, pengelolaan data lahan, pemesanan, penjemputan, dan akses nota transaksi dievaluasi untuk menilai keamanan komunikasi, perilaku runtime, serta integritas interaksi aplikasi dengan *API backend*.

1) Pengujian Keamanan *Transport Layer*



Gambar 6 *TLS/SSL tester result*

Gambar 6 menampilkan hasil pengujian lapisan transport menggunakan *TLS/SSL Security Tester*. Seluruh komunikasi antara aplikasi dan server menggunakan protokol *HTTPS* tanpa *cleartext HTTP*, dan uji *Cleartext Traffic*, *TLS Misconfiguration*, serta *Certificate Transparency Validation* memperoleh status lulus. Temuan ini mengindikasikan konfigurasi sertifikat *server* dan *protokol enkripsi* telah memenuhi standar keamanan dasar untuk menjaga *confidentiality* dan i*ntegrity* data selama transmisi. Namun, instrumentasi *Frida* menunjukkan aplikasi belum menerapkan mekanisme *SSL Pinning*, ditandai dengan keluaran "*CertificatePinner not found*".

```
Loaded Frida Script - ssl_pinning_bypass
[SSL Pinning Bypass] okhttp CertificatePinner not found
[SSL Pinning Bypass] okhttp3 CertificatePinner not found
[SSL Pinning Bypass] Comprehensive okhttp3 bypass not available
[SSL Pinning Bypass] okhttp3 ConnectionSpec not found
[SSL Pinning Bypass] DataTheorem trustkit not found
[SSL Pinning Bypass] Appcelerator PinningTrustManager not found
[SSL Pinning Bypass] Apache Cordova SSLCertificateChecker not found
[SSL Pinning Bypass] Wultra CertStore.validateFingerprint not found
```

Gambar 7 *Frida SSL pinning bypass*

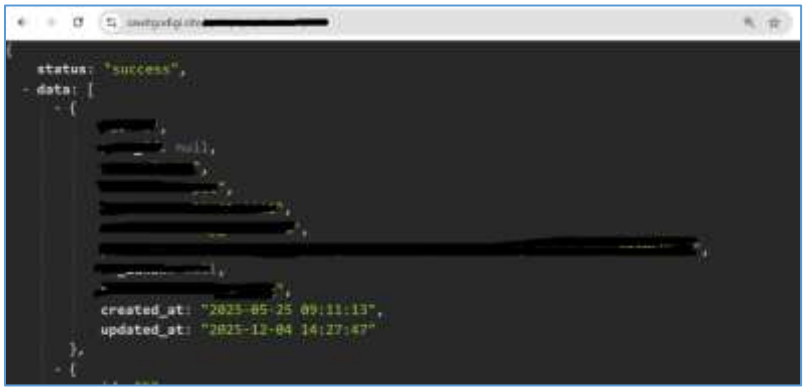
Gambar 7 menampilkan ketiadaan *SSL Pinning*, yang ditandai dengan keluaran *Frida* berupa pesan *"CertificatePinner not found"*, yang menyebabkan aplikasi tetap menerima koneksi meskipun terjadi penggantian sertifikat oleh pihak ketiga. Dengan demikian, meskipun kanal komunikasi telah terenkripsi, aplikasi masih rentan terhadap serangan *man-in-the-middle* pada jaringan publik atau jaringan yang telah disusupi. Penerapan *SSL Pinning* direkomendasikan sebagai penguatan tambahan pada lapisan *transport* untuk mencegah pencegatan dan modifikasi lalu lintas secara tidak sah.

2) Analisis *Endpoint* dan Validasi *API backend*

Analisis terhadap *trafik HTTP(S)* menunjukkan bahwa aplikasi mengakses sejumlah *endpoint backend* yang menangani berbagai fungsi, seperti unggah foto, pemuatan data profil, data lahan, serta informasi agen, petani, dan supir. Seluruh *URL endpoint* tersebut diperoleh dari hasil report analisis dinamis *MobSF*. Seluruh permintaan dikirim menggunakan metode *GET* tanpa dilengkapi token autentikasi ataupun parameter otorisasi lainnya. *Endpoint–endpoint* ini secara konsisten mengembalikan data dalam format *JSON*, sebagaimana ditunjukkan pada Gambar 8, yang

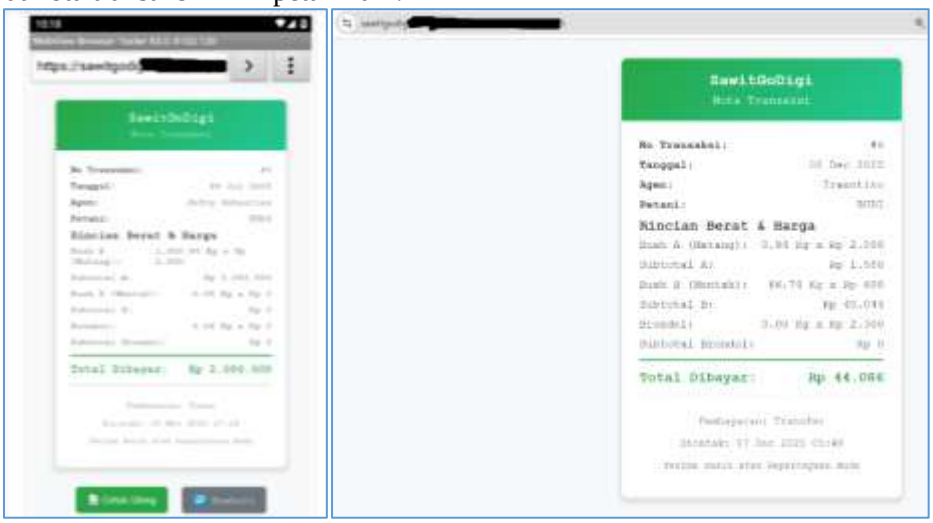
 ISSN:2302-8149
e-ISSN:2540-9719

menampilkan contoh keluaran *JSON* dari salah satu *endpoint* profil yang digunakan untuk mengambil data petani.



Gambar 8 Tampilan data *JSON* dari *endpoint API*

Berbeda dari *endpoint JSON* lainnya, *endpoint* yang digunakan untuk menampilkan nota transaksi menghasilkan halaman nota dalam bentuk tampilan siap cetak, sebagaimana terlihat pada Gambar 9. Secara normal, fitur ini hanya dapat diakses melalui menu riwayat transaksi setelah petani melakukan login. Namun, hasil pengujian menunjukkan bahwa halaman nota tersebut tetap dapat diakses melalui *URL* eksternal tanpa autentikasi, dan parameter identitas transaksi (ID) dapat diubah secara bebas melalui browser. Kondisi ini memungkinkan pengguna yang tidak memiliki hak akses untuk melihat nota transaksi milik petani lain.



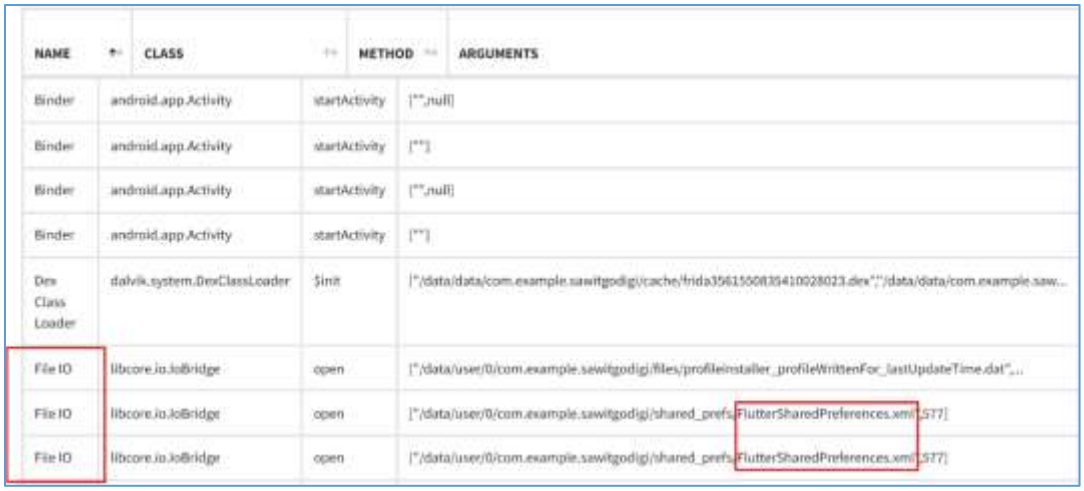
Gambar 9 Tampilan nota transaksi dari *endpoint* nota

Meskipun seluruh *endpoint* tersebut dirancang untuk dipanggil secara internal oleh aplikasi *mobile*, hasil pengujian menunjukkan bahwa *endpoint–endpoint* tersebut tetap dapat diakses secara langsung melalui browser tanpa *autentikasi*. Kondisi ini menegaskan adanya kelemahan kontrol akses pada sisi server, yang memunculkan kerentanan *Insecure Direct Object Reference (IDOR)* dan *Broken Access Control*, karena tidak terdapat verifikasi kepemilikan data sebelum informasi ditampilkan [21][23].

3) Analisis Perilaku *Runtime*

Gambar 10 menunjukkan hasil pemantauan perilaku *runtime* menggunakan *API Monitor* dan *Frida*. Aktivitas yang tercatat didominasi oleh operasi *File I/O* pada berkas internal aplikasi, termasuk akses berulang terhadap *FlutterSharedPreferences.xml*, pemanggilan *startActivity* pada komponen *android.app.Activity*, serta pemuatan kelas melalui *DexClassLoader*.

 ISSN:2302-8149
e-ISSN:2540-9719



Gambar 10 *API monitor MobSF*

Seluruh aktivitas tersebut berada dalam konteks penggunaan normal aplikasi dan tidak menunjukkan akses ke sumber daya sensitif seperti SMS, kontak, lokasi, maupun identitas perangkat. Temuan ini mengindikasikan bahwa dari sisi perilaku runtime lokal, aplikasi tidak menampilkan aktivitas yang bersifat mencurigakan atau berpotensi membahayakan pengguna.

4) *Logging* dan *Exception Handling*

Berdasarkan *Logcat Runtime Logs* yang ditunjukkan pada Gambar 11, tidak ditemukan pencatatan data sensitif seperti kata sandi, token autentikasi, atau isi respon *API* ke dalam *log* sistem.

```
pid=1378); deadline exceeded.
11-30 10:10:26.334   720  5463 E ActivityManager: ANR in com.example.sawitgodigi
(com.example.sawitgodigi/.MainActivity)
11-30 10:10:26.334   720  5463 E ActivityManager: PID: 5125
11-30 10:10:26.334   720  5463 E ActivityManager: Reason: Input dispatching timed out (7c7011b
com.example.sawitgodigi/com.example.sawitgodigi.MainActivity (server) is not responding. Waited
5002ms for FocusEvent(hasFocus=false))
11-30 10:10:26.334   720  5463 E ActivityManager: Parent: com.example.sawitgodigi/.MainActivity
```

Gambar 11 Logcat runtime Logs

Pesan yang terekam umumnya berupa informasi *debug* standar dan status eksekusi aplikasi. Beberapa catatan *Application Not Responding (ANR)* dan crash muncul selama pengujian, namun konteksnya mengarah pada keterbatasan lingkungan *emulator*, bukan pada adanya eksploitasi atau perilaku anomali yang memengaruhi keamanan data.

Tabel 4 merangkum hasil analisis dinamis yang menunjukkan bahwa aplikasi SawitGoDigi.apk versi 1.0.0 relatif aman pada sisi klien, namun ada kelemahan kritis pada sisi server. Seluruh komunikasi jaringan telah terenkripsi melalui *HTTPS*, aktivitas *runtime* berada dalam batas normal, dan tidak ditemukan pencatatan data sensitif pada *Logcat*. Hal ini mengindikasikan bahwa aplikasi tidak menunjukkan perilaku berbahaya dan tidak mengekspos data melalui mekanisme internal perangkat. Namun sejumlah *endpoint backend* dapat diakses tanpa autentikasi dan tidak memverifikasi parameter identitas, sehingga memungkinkan akses tidak sah terhadap data pengguna. Kondisi ini menghasilkan kerentanan *Broken Access Control, Insecure Direct Object Reference (IDOR)*, serta potensi paparan data sensitif, terutama pada *endpoint* nota transaksi yang dapat diakses hanya dengan memodifikasi nilai ID [20], [23].

Tabel 4 Ringkasan hasil analisis dinamis *mobsf*

Aspek Pengujian	Temuan Utama	Dampak / Risiko	Status Keamanan	Rekomendasi
Transport Layer Security (TLS)	Seluruh trafik menggunakan *HTTPS; TLS valid*; *Certificate Transparency* lulus. *SSL Pinning* tidak diterapkan.	*Best Practice Enhancement*; rentan *MITM* jika jaringan disusupi.	Aman bersyarat	Terapkan *SSL Pinning* pada aplikasi.
Endpoint Backend (Profil, Lahan, Agen, Supir,	Semua endpoint mengembalikan *JSON*; dapat diakses tanpa autentikasi; tidak	*Broken Access Control, IDOR, Sensitive Data*	Rentan	Terapkan autentikasi, otorisasi, dan

 ISSN:2302-8149
e-ISSN:2540-9719

Aspek Pengujian	Temuan Utama	Dampak / Risiko	Status Keamanan	Rekomendasi
Upload Foto)	ada verifikasi parameter identitas.	*Exposure*		validasi parameter.
Endpoint Nota Transaksi	Halaman nota dapat diakses publik dengan mengganti parameter ID; tidak memeriksa kepemilikan data.	*IDOR, Broken Access Control, Data Exposure*	Sangat rentan	Batasi akses berdasarkan identitas; validasi ID; wajib login.
Perilaku Runtime (*API Monitor & Frida*)	Aktivitas normal: *File I/O, SharedPreferences, startActivity, DexClassLoader*. Tidak ada API sensitif.	Tidak ada risiko keamanan langsung.	Aman	–
Logging dan *Exception Handling*	Tidak mencatat *password/token/respon API; ANR* muncul karena *emulator*.	Tidak ada risiko keamanan.	Aman	–

Dengan demikian, risiko keamanan utama pada aplikasi ini berasal dari desain *API backend* yang belum dilengkapi autentikasi, otorisasi, dan validasi permintaan.

4.5 Evaluasi Hasil Temuan

Analisis statis menunjukkan bahwa kelemahan utama aplikasi bersumber dari konfigurasi *build-time*, seperti penggunaan *debug certificate*, mode *debugging* aktif, *minSdkVersion* rendah, serta komponen yang diekspor tanpa proteksi. Perbaikan konfigurasi tersebut terbukti efektif meningkatkan keamanan struktural aplikasi, tercermin dari peningkatan *App Security Score* dari 43/100 (*Medium Risk*) menjadi 67/100 (*Low Risk*) dan hilangnya seluruh temuan berisiko *High*. Peningkatan ini menegaskan bahwa rekomendasi *MobSF* pada tahap statis relevan sebagai *baseline* untuk memperbaiki kelemahan teknis yang dapat dieksploitasi melalui *reverse engineering*, *debugging*, maupun penyalahgunaan komponen aplikasi.

Namun demikian, hasil analisis dinamis menunjukkan bahwa capaian kategori *Low Risk* pada analisis statis tidak secara otomatis menjamin keamanan aplikasi secara menyeluruh. Sejumlah *endpoint backend* masih dapat diakses tanpa autentikasi dan tanpa validasi hak akses, sehingga memunculkan kerentanan *Broken Access Control* dan *Insecure Direct Object Reference (IDOR)* [20], [23]. Temuan ini menunjukkan adanya kesenjangan antara keamanan sisi klien (*APK*) dan keamanan sisi server (*API*). Meskipun aplikasi tampak aman secara struktural, risiko kebocoran data tetap muncul melalui akses langsung ke *backend*. Selain itu, meskipun seluruh komunikasi jaringan telah menggunakan *HTTPS*, ketiadaan *SSL Pinning* menempatkan aplikasi pada kondisi aman bersyarat. Pada jaringan yang tidak tepercaya, aplikasi masih berisiko mengalami serangan *man-in-the-middle* apabila sertifikat perantara berhasil disisipkan. Karena itu, penguatan keamanan perlu dilakukan secara berlapis pada sisi klien dan server.

Secara komparatif, temuan penelitian ini sejalan dengan penelitian [15] dan [11] yang juga melaporkan kelemahan konfigurasi internal serta *bypass* terhadap mekanisme *SSL Pinning* pada tahap analisis dinamis. Pola serupa terlihat pada penelitian [16], yang mengidentifikasi lemahnya kontrol akses dan risiko data *exposure* pada sisi server. Sementara itu, penelitian [17] menunjukkan peningkatan skor keamanan setelah perbaikan statis, namun tidak mengidentifikasi kerentanan pada *backend* karena tidak dilakukan analisis dinamis. Dibandingkan penelitian-penelitian tersebut, penelitian ini menekankan pentingnya evaluasi mitigasi secara bertahap yang divalidasi ulang melalui analisis dinamis untuk memperoleh gambaran keamanan aplikasi yang lebih komprehensif.

4.6 Implikasi Penelitian

Dari sisi implikasi praktis, hasil penelitian ini menunjukkan bahwa peningkatan *App Security Score* perlu dipahami sebagai indikator perbaikan keamanan struktural, bukan sebagai jaminan keamanan *end-to-end*. Pengembang aplikasi perlu memprioritaskan penguatan keamanan *backend* melalui penerapan autentikasi berbasis token, mekanisme otorisasi yang sesuai, serta validasi parameter dan kepemilikan data pada setiap endpoint. Dalam konteks operasional, lemahnya kontrol akses backend berpotensi memungkinkan pihak tidak berwenang mengakses data transaksi dan data operasional. Kondisi ini dapat memicu kebocoran informasi bisnis serta menurunkan kepercayaan pengguna terhadap sistem. Oleh karena itu, bagi organisasi atau pengelola sistem, penguatan kontrol

 ISSN:2302-8149
e-ISSN:2540-9719

akses serta penerapan mekanisme pemantauan aktivitas *API* menjadi prasyarat penting sebelum aplikasi diterapkan pada lingkungan operasional.

Secara metodologis, penelitian ini menunjukkan bahwa penggunaan *MobSF* akan lebih efektif apabila diterapkan sebagai alur pengujian bertahap, dimulai dari analisis statis untuk perbaikan konfigurasi dasar dan dilanjutkan dengan analisis dinamis untuk memvalidasi perilaku *runtime* serta interaksi dengan *backend* melalui pengamatan interaksi *runtime* dan *endpoint API*, sebagaimana juga ditekankan dalam studi analisis keamanan *API mobile* berbasis pendekatan *programatik* [24]. Pendekatan ini memungkinkan pengujian keamanan yang lebih menyeluruh dengan mencakup sisi klien dan server secara bersamaan.

5 Kesimpulan

Penelitian ini menunjukkan bahwa penerapan analisis statis dan dinamis pada *MobSF* efektif untuk mengevaluasi keamanan aplikasi *mobile* Sistem Informasi Panen Sawit(sawitgodigi.apk versi 1.0.0). Analisis statis mengidentifikasi sejumlah kerentanan konfigurasi penting seperti penggunaan sertifikat *debug*, mode *debugging* aktif, dan komponen yang diekspor tanpa proteksi. Setelah dilakukan perbaikan teknis, seluruh temuan berisiko tinggi berhasil dihilangkan dan *App Security Score* meningkat dari 43/100 menjadi 67/100. Analisis dinamis membuktikan bahwa aplikasi telah aman pada sisi klien, ditandai dengan komunikasi *HTTPS* yang terenkripsi, perilaku *runtime* normal, dan tidak adanya pencatatan data sensitif. Namun, pengujian juga mengungkap kelemahan kritis pada sisi server, yaitu *endpoint backend* yang dapat diakses tanpa autentikasi dan validasi hak akses, sehingga menimbulkan risiko *Broken Access Control* dan *IDOR*. Dengan demikian, penelitian ini menegaskan bahwa meskipun aplikasi telah mencapai keamanan struktural melalui perbaikan statis, keamanan keseluruhan tetap bergantung pada penguatan *API backend*. Pendekatan dua tahap ini memberikan gambaran keamanan yang lebih komprehensif dan menunjukkan pentingnya mitigasi yang mencakup sisi klien dan server secara bersamaan sebelum aplikasi diterapkan pada lingkungan operasional. Penelitian ini memiliki keterbatasan karena belum mencakup analisis langsung terhadap kode sumber *backend* maupun pengujian penetrasi secara menyeluruh. Penelitian selanjutnya disarankan untuk memperluas pengujian keamanan *API backend* dengan mengacu pada standar seperti *OWASP MASVS* dan *OWASP Top 10 API Security Risks*.

Referensi

[1] DataIndonesia.id, "Deretan Negara dengan Persentase Pengguna Internet Melalui Ponsel Terbesar pada Kuartal IV/2024," 2024. Accessed: May 30, 2025. [Online]. Available: 1. https://dataindonesia.id/internet/detail/deretan-negara-dengan-persentase-pengguna-internet-melalui-ponsel-terbesar-pada-kuartal-iv2024

[2] Asosiasi Penyelenggara Jasa Internet Indonesia (APJII), "Laporan Survei Internet Indonesia 2024," Jakarta, 2024. Accessed: May 30, 2025. [Online]. Available: https://apjii.or.id/

[3] A. Ardi, Y. Aufar, A. Eko Syaputra, and M. Saputra, "*Integration of Geolocation and Real-Time Data for Optimizing BoardingHouse Search in Mobile Applications*," *Sistemasi: Jurnal Sistem Informasi*, Vol. 14, No. 3, pp. 2540–9719, Mar. 2025.

[4] DataIndonesia.id, "Data Volume Ekspor Minyak Sawit Indonesia 1 Tahun Terakhir hingga Januari 2025," 2025. Accessed: May 30, 2025. [Online]. Available: 2. https://dataindonesia.id/industri-perdagangan/detail/data-volume-ekspor-minyak-sawit-indonesia-1-tahun-terakhir-hingga-januari-2025

[5] M. V. Alfarisi and D. Enda, "Rancang Bangun Aplikasi berbasis *Mobile* untuk Perhitungan dan Pelaporan pada Agen Penimbangan Sawit," *Edukasi Terkini: Jurnal Pendidikan Modern*, Vol. 7, No. 3, pp. 65–77, Sep. 25AD.

[6] N. Hidayasari, Kasmawi, Mansur, P. Nuranisa, and M. Iqbal Husaini, "Analisis Penerapan Aspek Keamanan Informasi CIA Triad pada Sistem Informasi Akademik," in *Seminar Nasional Industri dan Teknologi (SNIT)*, Bengkalis: Politeknik Negeri Bengkalis, Nov. 2024, pp. 90–96.

[7] Zimperium, "*2025 Global Mobile Threat Report*," 2025. Accessed: May 30, 2025. [Online]. Available: https://zimperium.com/gmtr-track-25

[8] A. Abraham, "*Mobile Security Framework (MobSF)*," GitHub. Accessed: Jun. 06, 2025. [Online]. Available: https://github.com/MobSF/Mobile-Security-Framework-MobSF

 ISSN:2302-8149
e-ISSN:2540-9719

[9] F. Nurindahsari and B. P. Zen, "Analisis Statik Keamanan Aplikasi Video Streaming berbasis *Android* menggunakan *Mobile Security Framework (Mobsf) Security Static Analysis of Android-based Video Streaming Application using Mobile Security Framework (Mobsf)*," *Cybersecurity and Digital Forensics Journal*, Vol. 4, No. 2, pp. 63–80, Apr. 2022.
[10] R. Abdillah, A. A. Trinoto, and I. Himawan, "*Static Analysis using Mobile Security Framework for Smart Home Appliances*," *Journal of Information System, Applied, Management, Accounting and Research*, Vol. 7, No. 3, pp. 760–765, Jul. 2023.
[11] S. U. Kusreynada and A. S. Barkah, "*Android Apps Vulnerability Detection with Static and Dynamic Analysis Approach using MOBSF*," *Journal of Computer Science and Engineering (JCSE)*, Vol. 5, No. 1, pp. 46–63, Apr. 2024, DOI: http://dx.doi.org/10.36596/jcse.v5i1.789.
[12] C. Anwar, C. Herli Sumerli A, N. Rahayu, and K. Kraugusteeliana, "*The Application of Mobile Security Framework (MOBSF) and Mobile Application Security Testing Guide to Ensure the Security in Mobile Commerce Applications*," *Jurnal Sistim Informasi dan Teknolog*, Vol. 5, No. 2, pp. 97–102, Jun. 2023, DOI: https://doi.org/10.37034/jsisfotek.v5i2.231.
[13] T. B. Subakja, M. Fronita, S. Syaifullah, T. Khairil Ahsyar, and S. Siregar, "Analisis Perbandingan Keamanan Apliksi Transportasi Online berbasis *Android* menggunakan *Mobile Security Framework (Mobsf)*," *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika*, Vol. 10, No. 2, pp. 1823–1837, Jul. 2025, DOI: https://doi.org/10.29100/jipi.v10i2.6185.
[14] F. Awanda Alviansyah and E. Ramadhani, "Implementasi *Dynamic Application Security Testing* pada Aplikasi berbasis *Android*," *Automata*, Vol. 2, No. 1, Jan. 2021.
[15] K. N. Isnaini and D. Suhartono, "*Security Analysis of Simpel Desa using Mobile Security Framework and ISO 27002:2013*," *INTENSIF: Jurnal Ilmiah Penelitian dan Penerapan Teknologi Sistem Informasi*, Vol. 7, No. 1, pp. 84–105, Feb. 2023.
[16] S. Erbeliza, "Analisis Keamanan Aplikasi *Mobile Commerce* menggunakan *Mobile Security Framework (Mobsf)* dan *Owasp Mobile Application Security Testing Guide (Mastg)*," Undergraduate thesis, Universitas Islam Negeri Syarif Hidayatullah Jakarta, Jakarta, 2023.
[17] P. Nur Izzati and K. Kasmawi, "*Static Analysis-based Security Enhancement for Mobile Applications using Mobile Security Framework (MOBSF)*," *Journal of Applied Informatics and Computing*, Vol. 9, No. 4, pp. 1272–1279, Aug. 2025, DOI: doi.org/10.30871/jaic.v9i4.9525.
[18] P. Rizkika, D. Juardi, A. Susilo Yuda Irawan Informatika, U. Singaperbangsa Karawang Jl HSRonggo Waluyo, T. Timur, and J. Barat, "Analisis Keamanan pada Aplikasi HIMFO berbasis *Android* menggunakan *Mobsf*," *JATI (Jurnal Mahasiswa Teknik Informatika)*, Vol. 8, No. 4, pp. 5945–5952, Aug. 2024.
[19] O. K. Syahputra, A. Rizki Jatmiko, and A. P. Sanusi, "Evaluasi Keamanan Aplikasi Jogo Malang Presisi dengan Metode *Mobile Security Framework (MOBSF)* melalui Analisis Statis," *Seminar Nasional Sistem Informasi*, Vol. 8, pp. 4796–4802, Dec. 2024.
[20] F. Tanveer, F. Iradat, W. Iqbal, and A. Ahmad, "*Towards Secure APIs: A Survey on RESTful API Vulnerability Detection*," *Computers, Materials and Continua*, Vol. 84, No. 3, pp. 4223–4257, Jul. 2025, DOI: https://doi.org/10.32604/cmc.2025.067536.
[21] A. S. Filho, R. J. Rodríguez, and E. L. Feitosa, "*Automated Broken Object-Level Authorization Attack Detection in REST APIs Through OpenAPI to Colored Petri Nets Transformation*," *Int J Inf Secur*, Vol. 24, No. 2, pp. 83–101, Apr. 2025, DOI: https://doi.org/10.1007/s10207-024-00970-5.
[22] Sancsoft, "*Best Practices for Static Analysis of Mobile Apps*," Sancsoft. Accessed: Jul. 19, 2025. [Online]. Available: https://www.sancsoft.com/resources/best-practices/static-analysis-for-mobile-apps/
[23] A. Anas, A. A. Alhelbawy, S. El Gamal, and B. Youssef, "*BACAD: AI-based Framework for Detecting Vertical Broken Access Control Attacks*," *Egyptian Informatics Journal*, Vol. 28, p. 100571, Dec. 2024, DOI: https://doi.org/10.1016/j.eij.2024.100571.
[24] N. Haris, K. Chen, A. Song, and B. Pou, "*Finding Vulnerabilities in Mobile Application APIs: A Modular Programmatic Approach*," *arXiv preprint*, Oct. 2023, DOI: https://doi.org/10.48550/arXiv.2310.14137.